Docket No.: 33227/217001; SUN040284
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Sunay Tripathi

Confirmation No.: 1500

Application No.: 10/698,212

Art Unit: 2451

Filed: October 31, 2003

Examiner: B. Tiv

For: METHODS AND APPARTUS FOR COORDINATING PROCESSING OF NETWORK CONNECTIONS BETWEEN TWO NETWORK PROTOCOL STACKS

APPEAL BRIEF

MS Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Madam:

Pursuant to 37 CFR § 41.37, please consider the following Appellant's Brief in the referenced application currently before the Board of Patent Appeals and Interferences ("the Board").

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Cases

Statutes

Other Authorities

I. REAL PARTY IN INTEREST

The real party in interest for this appeal is Sun Microsystems, Inc. An Assignment transferring all interest in the referenced application from the inventors to Sun Microsystems, Inc was recorded by the USPTO on October 31, 2003. The Assignment is recorded at Reel 014659, Frame 0265.

II. RELATED APPEALS AND INTERFERENCES

To the best of the knowledge of the Appellants and Appellants' legal representative, there are no other appeals, interferences, or judicial proceedings which will directly affect or be directly affected by, or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

A. Total Number of Claims in Application

There are 31 claims pending in application.

B. Current Status of Claims

1. Claims canceled:
6-8, 12, 14-16, 18, 20-23, 27, 29, 34-35, 39, 41-44, 46, 48-51, and 55

2. Claims pending:
1-5, 9-11, 13, 17, 19, 24-26, 28, 30-33, 36-38, 40, 45, 47, 52-54, and 56-58

3. Claims rejected:
1-5, 9-11, 13, 17, 19, 24-26, 28, 30-33, 36-38, 40, 45, 47, 52-54, and 56-58

C. Claims On Appeal

The claims on appeal are claims 1-5, 9-11, 13, 17, 19, 24-26, 28, 30-33, 36-38, 40, 45, 47, 52-54, and 56-58.

IV. STATUS OF AMENDMENTS

Applicant did not file an Amendment After Final Rejection. Thus, all of the amendments have been entered and considered by the Examiner. The pending claims of record are presented in the Claims Appendix. The claims in the Claims Appendix include the amendments filed by the Applicant on October 2, 2008.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The following discussion summarizes the content of the claimed subject matter. The references to the Figures and Specification included below should not be construed as the only locations in the specification which support or discuss the respective limitation.

Turning to the claims, independent claim 1 is directed toward a method for processing a network connection on a computer system. *See* Specification: page 4 lines 1-15. The method involves establishing a network connection by a software network protocol stack. *Id*: page 20 lines 5-19. The software network protocol stack is implemented in the kernel of an operating system associated with a computer system. *Id*: page 12 lines 11-19. The kernel maintains kernel-level connection state information for the network connection. *Id*: page 17 lines 4-17. A socket layer maintains socket layer-level connection state information for the network connection. *Id*: page 16 line 15 – page 17 line 4. The method involves determining whether to offload the network connection from the software network protocol stack to a hardware network protocol stack. *Id*: page 20 line 16 – page 21 line 4. The hardware network protocol stack is implemented in a TCP Offload Engine (TOE)-capable network interface card operatively connected to the computer system. *Id*: page 20 lines 5-8; page 14 line 24 – page 15 line 1. The method involves transferring the network connection from the software network protocol stack to the hardware network protocol stack using a

network interface card driver when it is determined to offload the network connection from the software network protocol stack to the hardware network protocol stack. *Id*: page 15 lines 3-15. The method involves determining to accept the transfer of the network connection at the hardware network protocol stack based on a processing capability of the hardware network protocol stack. *Id*: page 30 line 21 – page 31 line 16. The network interface card maintains hardware-level connection state information for the network connection. *Id*: page 17 line 18 – page 18 line 2. After the hardware network protocol stack accepts transfer of the network connection, the software network protocol stack is configured to continually reference hardware-level connection state information and the hardware network protocol stack is configured to continually reference kernel-level connection state information and socket layer-level connection state information. *Id*: page 16 line 11 – page 17 line 3; and page 22 lines 3-15.

Independent claim 30 is directed toward an apparatus for processing a network connection in a computer system. *Id*: page 4 lines 1-15. The apparatus involves means for establishing the network connection by a software network protocol stack. The means for establishing the network connection is the software network protocol stack as shown, for example, by element 210 of Fig. 2 executing on the physical hardware shown in Fig. 11. *Id*: page 20 lines 5-19. The software network protocol stack is implemented in the kernel of an operating system associated with a computer system. *Id*: page 12 lines 11-19. The kernel maintains kernel-level connection state information for the network connection. *Id*: page 17 lines 4-17. A socket layer maintains socket layer-level connection state information for the network connection. *Id*: page 16 line 15 – page 17 line 4. The apparatus further involves means for determining whether to offload the network connection from the software network protocol stack to a hardware network protocol stack. The means for

determining whether to offload the network connection may be performed, for example, by the operating system kernel as shown by element 202 of Fig. 2, by the socket layer of the operating system kernel as shown by element 204 of Fig. 2, or by the software network protocol stack as shown by element 210 of Fig. 2 all executing on the physical hardware shown in Fig. 11. *Id*: page 20 line 16 – page 21 line 4. The hardware network protocol stack is implemented in a TOE-capable network interface card operatively connected to the computer system. *Id*: page 20 lines 5-8; page 14 line 24 – page 15 line 1. The apparatus further involves means for transferring the network connection from the software network protocol stack to the hardware network protocol stack when it is determined to offload the network connection. The means for transferring the network connection may be the TOE capable NIC software driver as shown by element 220 of Fig. 2 executing on the physical hardware shown in Fig. 11. *Id*: page 15 lines 3-15. The apparatus further involves means for determining to accept the transfer of the network connection at the hardware network protocol stack based on a processing capability of the hardware network protocol stack. The means for determining to accept the transfer of the network connection may be the hardware network protocol stack as shown by element 216 of Fig. 2 executing on the physical hardware shown in Fig. 11. *Id*: page 30 line 21 – page 31 line 16. The network interface card maintains hardware-level connection state information for the network connection. *Id*: page 17 line 18 – page 18 line 2. After the hardware network protocol stack accepts transfer of the network connection, the software network protocol stack is configured to continually reference hardware-level connection state information and the hardware network protocol stack is configured to continually reference kernel-level connection state information and socket layer-level connection state information. *Id*: page 16 line 11 – page 17 line 3; and page 22 lines 3-15.

Independent claim 31 is directed toward a computer-readable medium storing thereon computer-readable instructions for processing a network connection in a computer system. *Id*: page 8 lines 18-22. The instructions involve establishing the network connection by a software network protocol stack. *Id*: page 20 lines 5-19. The network protocol stack is implemented in the kernel of an operating system associated with a computer system. *Id*: page 12 lines 11-19. The kernel maintains kernel-level connection state information for the network connection. *Id*: page 17 lines 4-17. A socket layer maintains socket layer-level connection state information for the network connection. *Id*: page 16 line 15 – page 17 line 4. The instructions involve determining whether to offload the network connection from the software network protocol stack to a hardware network protocol stack. *Id*: page 20 line 16 – page 21 line 4. The hardware network protocol stack is implemented in a TOE-capable network interface card operatively connected to the computer system. *Id*: page 20 lines 5-8; page 14 line 24 – page 15 line 1. The instructions involve transferring the network connection from the software network protocol stack to the hardware network protocol stack using a network interface card driver when it is determined to offload the network connection. *Id*: page 15 lines 3-15. The instructions involve determining to accept the transfer of the network connection at the hardware network protocol stack based on a processing capability of the hardware network protocol stack. *Id*: page 30 line 21 – page 31 line 16. The network interface card maintains hardware-level connection state information for the network connection. *Id*: page 17 line 18 – page 18 line 2. After the hardware network protocol stack accepts transfer of the network connection, the software network protocol stack is configured to continually reference hardware-level connection state information and the hardware network protocol stack is configured to continually reference

kernel-level connection state information and socket layer-level connection state information. *Id*: page 16 line 11 – page 17 line 3; and page 22 lines 3-15.

Independent claim 32 is directed toward a network device. *Id*: page 4 line 16 – page 5 line 3. The network device includes an operating system associated with a computer system. *Id*: page 12 lines 11-19. A software network protocol stack is implemented in a kernel of the operating system. *Id*. The kernel maintains kernel-level connection state information for a network connection. *Id*: page 17 lines 4-17. A socket layer maintains socket layer-level connection state information for the network connection. *Id*: page 16 line 15 – page 17 line 4. The network device further includes a hardware network protocol stack coupled to the software network protocol stack. *Id*: page 20 line 16 – page 21 line 4. The hardware network protocol stack is implemented in a TOE-capable network interface card operatively connected to the computer system. *Id*: page 20 lines 5-8; page 14 line 24 – page 15 line 1. The operating system is configured to determine whether to offload the network connection to the hardware network protocol stack. *Id*: page 20 line 16 – page 21 line 4. When the operating system determines to offload the network connection, the operating system is further configured to transfer the network connection from the software network protocol stack to the hardware network protocol stack using a network interface card driver. *Id*: page 15 lines 3-15. The network device further includes a control component which is configured to determine acceptance of the transfer of the network connection at the hardware network protocol stack based on a processing capability of the hardware network protocol stack. *Id*: page 30 line 21 – page 31 line 16. The network interface card maintains hardware-level connection state information for the network connection. *Id*: page 17 line 18 – page 18 line 2. After the hardware network protocol stack accepts transfer of the network connection, the software network protocol stack is configured to continually

reference hardware-level connection state information and the hardware network protocol stack is configured to continually reference kernel-level connection state information and socket layer-level connection state information. *Id*: page 16 line 11 – page 17 line 3; and page 22 lines 3-15.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL[1]

The present Appeal addresses the following ground of rejection:

A. Whether claims 1-5, 9-11, 13, 17, 19, 24-26, 28, 30-33, 36-38, 40, 45, 47, 52-54, and 56-58 are properly rejected under 35 U.S.C. § 112, first paragraph.

B. Whether claims 1-5, 9-11, 13, 17, 19, 24, 28, 30-33, 36-38, 40, 45, 47, 52, and 56-58 are patentable under 35 U.S.C. § 103 over U.S. Patent Application No. 2004/0064589 ("Boucher") and U.S. Patent No. 6,697,868 ("Craft"). For purposes of this Appeal, claims 1-5, 9-11, 13, 17, 19, 24, 28, 30-33, 36-38, 40, 45, 47, 52, and 56-58 stand or fall together. Independent claim 1 is representative of the group including claims 1-5, 9-11, 13, 17, 19, 24, 28, 30-33, 36-38, 40, 45, 47, 52, and 56-58.

VII. ARGUMENT

A. Examiner has made an improper rejection under 35 U.S.C. § 112, first paragraph

Claims 1-5, 9-11, 13, 17, 19, 24-26, 28, 30-33, 36-38, 40, 45, 47, 52-54, and 56-58 stand rejected under 35 U.S.C. § 112, first paragraph, as failing to comply with the written description requirement. *See* page 3 of the Office Action. In making the rejection, the Examiner asserts that "the claims contain subject matter which was not described in the specification in such a way as to

[1] Appellants will address the provisional double patenting rejection, to the extent it is still applicable, upon resolution of the instant appeal. Further, Appellants will address the Examiner's claim objection on page 2 of the Final Office Action dated December 18, 2008, upon resolution of the instant appeal.

reasonably convey to one skilled in the relevant art that the inventors, at the time the application was filed, had possession of the claimed invention." *Id.* Specifically, in referencing the limitation:

> wherein after the hardware network protocol stack accepts transfer of the network connection the software network protocol stack is configured to continually reference hardware-level connection state information and the hardware network protocol stack is configured to continually reference kernel-level connection state information and socket layer-level connection state information

the Examiner submits that the feature common to both the hardware and software network protocol stacks of *continually referencing* connection state information maintained by the other is not found in the specification. *See* pages 3-4 of the Office Action. Appellants disagree and assert that the pending claims satisfy 35 U.S.C. § 112, ¶1 for at least the reasons set forth below.

The inventors were in possession of the specific subject matter claimed as of the relied-upon filing date of the application

The first paragraph of 35 U.S.C. § 112 states that "the specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention." In particular, the written description requirement of first paragraph of 35 U.S.C. § 112 is directed to ensuring that the inventor had possession of, as of the filing date of the application relied on, the specific subject matter later claimed by the inventor. *See* MPEP § 2161.01 (citing *In re Herschler*, 591 F.2d 693, 700-01 (CCPA 1979) and *In re Kaslow,* 707 F.2d 1366, 217 (Fed.Cir. 1983)).

Turning to the rejection, Appellants respectfully assert that limitation recited by the pending claims that serves as the basis for the Examiner's rejection is in fact found in the specification. For example, connection state information may be maintained in support of two different network protocol stacks. *See* page 16 lines 7-8 of the specification. Specifically, connection state information may be maintained by the socket layer, the kernel, and the TOE capable NIC. *See* page 16 lines 9-11 of the specification. Access to the connection state information may be facilitated between the hardware and software network protocol stacks by passing a *pointer*. *See* page 16 lines 11-14 of the specification. Appellants respectfully assert it is well known in the related art that a pointer is a reference to a particular location in memory where data is stored. Accordingly, the pointer may be used at any time to reference, access, and possibly modify that data item in its current and most relevant state. Because the accessibility of connection state information between the software and hardware network protocol stacks is facilitated by use of pointers, it follows that the software network protocol stack, which possesses a pointer to the hardware-level connection state information, is configured to continually reference the hardware-level connection state information. Similarly, the hardware network protocol stack, which possesses pointers to both the kernel-level and socket layer-level connection state information, is able to continually reference both the kernel-level and socket layer-level connection state information. In view of the above, it is clear that the inventors had possession of, as of the filing date of the application relied on, the specific subject matter (*i.e.,* the subject matter related to the "continually referencing" aspect to the invention) later claimed by the inventor.

There is no requirement that the exact term used in the language of the claim must appear in the specification

37 C.F.R. § 1.121(e) states a requirement of "*substantial correspondence* between the claims, the remainder of the specification, and the drawings." MPEP § 1302.01 (citing 37 § C.F.R. 1.121(e)). Further, MPEP § 1302.01 acknowledges that, while language of originally-filed claims typically follow nomenclature of the specification, sometimes during the course of making amendments terms are employed which do not appear in the specification. *Id.* In such instances, "*exact terms need not be used* … to satisfy the written description requirement of the first paragraph of 35 U.S.C. § 112." *Id* (citing *Eiselstein v. Frank*, 52 F.3d 1035, 1038 (Fed. Cir. 1995); *In re Wertheim*, 541 F.2d 257, 265 (CCPA 1976)).

Because the standard for specification-claim consistency is substantial correspondence rather than exact term usage, Appellants respectfully assert that the "continually referenced" language, as discussed above and recited in the independent claims, satisfies the standard as stated in 37 C.F.R. § 1.121(e) for reasons already discussed above in § VII(A)(1). In particular, because both the software network protocol stack and the hardware protocol stack include a pointer to the appropriate connection state information, both the software network protocol stack and the hardware protocol stack may be said to "continually reference" the appropriate connection state information (via the appropriate pointers) for at least the reasons discussed above in § VII(A)(1).

Summary

For at least these reasons, Appellants respectfully assert that the limitation of "continuously referencing…", as discussed above and recited by the pending claims, is supported by the

originally-filed specification. Accordingly, the inventors are ensured of having possession of, as of the filing date of the application relied on, the specific subject matter later claimed by the inventors under the meaning of 35 U.S.C. § 112, first paragraph. As such, the rejection under 35 U.S.C. § 112, first paragraph is improper and should be withdrawn.

B. Examiner has failed to provide sufficient evidence of prima facie obviousness with respect to claims 1-5, 9-11, 13, 17, 19, 24, 28, 30-33, 36-38, 40, 45, 47, 52, and 56-58

The Examiner has failed to show sufficient evidence to establish a *prima facie* case of obviousness with respect to the pending claims. Specifically, the Examiner has misconstrued the phrase "hardware network protocol stack" and further mischaracterized both Boucher and Craft as disclosing or rendering obvious a hardware network protocol stack and a software network protocol stack. Further, the Examiner has also effectively read out the limitation recited by the pending claims of "the software network protocol stack is configured to continually reference hardware-level connection state information and the hardware network protocol stack is configured to continually reference kernel-level connection state information and socket layer-level connection state information." For at least these reasons, the Examiner's rejection should be reversed.

The Examiner has misconstrued the phrase "hardware network protocol stack" in the claims

Appellants assert that the Examiner has misconstrued the phrase "hardware network protocol stack" in maintaining the rejection. In particular, the Examiner has disregarded the requirements set out by *Phillips* to determine meaning of a claim term. Accordingly, the underlying basis for the Examiner's rejection is improper.

Specifically, *Phillips* requires the Examiner to read the claimed limitations in light of the Specification. In particular, "[t]he person of ordinary skill in the art is deemed to read the claim term not only in the context of the particular claim in which the disputed term appears, but *in the context of the entire patent, including the specification*." *See Phillips v. AWH Corp.*, 415 F.3d 1303, 1313 (Fed. Cir. 2005) (en banc) (emphasis added). Based on *Phillips* and as clearly described in the Specification, a *network protocol stack* is a set of network protocol layers (*i.e.*, a network protocol layer is a distinct program or process, such as TCP, IP, etc.) through which data must pass for processing during a data exchange over a network. *See* specification: page 2 lines 1-8. Further, a *hardware* network protocol stack, as recited in the claims, is a network protocol stack that is implemented in a TCP Offload Engine (TOE)-capable network interface card (NIC). *See* specification: page 14 line 22 – page 15 line 2. Contrary to *Phillips,* the aforementioned definition of hardware network protocol stack was not used by the Examiner as evidenced by the Examiner's rejection of the pending claims.

Boucher fails to disclose or render obvious a hardware network protocol stack

The Examiner cites to Boucher, in part, to disclose a hardware network protocol stack. *See* Office Action: pages 6-7 (paragraphs beginning "determining whether to offload…", "transferring the network connection…", and "wherein the network interface card…"). Boucher is directed to an intelligent network interface card (INIC) / communication processing device (CPD) that is operatively connected to a host. *See* Boucher: Fig. 1. In receiving a stream of packets, the CPD processes each incoming packet in part by determining whether processing should use a slow-path

or a fast-path processing mechanism. *See* Boucher: paragraphs [0052] and [0053]. A slow-path processing mechanism involves sending the packet to the host for processing by the host protocol stack. *See* Boucher: paragraph [0060]. Once a packet from a connection has been slow-path processed by the host protocol stack, a communication control block (CCB) is created and sent to the INIC/CPD for storage. *Id.* Use of the CCB by the INIC/CPD allows for subsequent packets from the same connection to be eligible for fast-path processing. *See* Boucher: paragraph [0061]. Fast-path processing allows a subsequent packet from the same connection to be stored directly on host and therefore bypass the session layer, transport layer, and network layer. *Id;* also see Fig. 6 of Boucher. In view of this, Appellants respectfully assert that Boucher at best merely contemplates a network protocol stack which resides upon the host and is used to process packets via the slow-path mechanism. See Boucher: Fig. 2 and Figs. 4a-5 (where the network protocol stack, *i.e.* element 44, exists outside of the INIC/CPD, *i.e.* element 30). Because this network protocol stack of Boucher does not operate on the INIC/CPD, the network protocol stack of Boucher cannot be properly characterized as a hardware network protocol stack as recited in claims and construed in view of the specification (as discussed above).

Further, Appellants argue in the alternative that there is no feature located on the INIC/CPD disclosed by Boucher which may be properly construed as a hardware network protocol stack. In fact, Boucher's fast-path processing mechanism allows selected packets to bypass the various network layer processing steps thereby accelerating overall network communication. *See* Boucher: paragraph [0061]; Fig. 6; and Abstract. Accordingly, Appellants respectfully assert that a network protocol stack (*i.e.*, requiring processing of data by each of the separate protocol layers) and the fast-path processing mechanism disclosed by Boucher (*i.e.*, bypassing processing of data by each of

the separate protocol layers for select packets) are each contrary to the other regarding functionality and provided benefit. Said another way, construing the INIC/CPD of Boucher to feature a hardware network protocol stack, as the Examiner suggests, defeats the very purpose of the INIC/CPD's fast-track processing mechanism (*i.e.,* the ability to bypass one or more of the protocol layers).

In view of arguments presented above, Appellants respectfully assert that the Examiner has mischaracterized Boucher as disclosing or rendering obvious limitations of independent claim 1 that recite a hardware network protocol stack. As such, Boucher cannot be properly relied upon to disclose or render obvious all the limitations of independent claim 1.

Craft fails to provide or otherwise render obvious that which Boucher lacks

The Examiner also cites to Craft, in part, to disclose a hardware network protocol stack. *See* Office Action: page 7 (paragraph beginning "Craft teaches offloading..."). Appellants respectfully assert that Craft is directed to an invention having very similar infrastructure to Boucher and therefore fails to provide for a hardware network protocol stack for the same or similar reasons discussed in relation to Boucher. *See* Craft: Figs. 1 and 2. As with Boucher, Craft discloses a network protocol stack that clearly exists apart from an INIC/CPD. *See* Craft: Fig. 1 (where the network protocol stack, *i.e.* element 44, exists outside of the INIC/CPD, *i.e.* element 30). Also, as with Boucher, Craft discloses, at best, a single network protocol stack running on a host CPU, which serves to create a communication context that is cached on the CCB. *See* Craft: column 3 lines 23-46. Another similarity with Boucher is Craft's distinction between a fast-path and a slow-path processing mechanisms for incoming which serve the same or similar functions to those

features as disclosed by Boucher. *See* Craft: column 3 lines 47-60; and column 4 line 41 – column 5 line 34. Because Craft can only be properly construed to disclose a software network protocol stack, Appellants respectfully assert that Craft cannot be properly characterized as disclosing a hardware network protocol stack for at least the same reasons discussed above in relation to Boucher.

In view of arguments presented above, Appellants respectfully assert that the Examiner has mischaracterized Craft as disclosing or render obvious limitations of independent claim 1 that recite a hardware network protocol stack. As such, Craft cannot be properly relied upon to disclose or render obvious all the limitations of independent claim 1.

The Examiner has effectively read out the limitation of "the software network protocol stack is configured to continually reference hardware-level connection state information and the hardware network protocol stack is configured to continually reference kernel-level connection state information and socket layer-level connection state information"

In making the rejection, the Examiner relies upon Boucher to disclose or render obvious "*wherein* the network interface card maintains hardware-level connection state information for the network connection, and *wherein* after the hardware network protocol stack accepts transfer of the network connection the software network protocol stack is configured to continually reference hardware-level connection state information and the hardware network protocol stack is configured to continually reference kernel-level connection state information and socket layer-level connection state information." *See* page 7 of the Office Action (paragraph beginning "wherein the network interface card maintains..."). In citing to Boucher, the Examiner cites to Figs. 22, 23, and 8 (by way of citing to paragraph [0068]). Appellants respectfully assert that the references to Figs. 8, 22,

and 23 of Boucher at best merely disclose "wherein the network interface card maintains hardware-level connection state information for the network connection." Consequently, the Examiner has read out the limitation of "wherein after the hardware network protocol stack accepts transfer of the network connection the software network protocol stack is configured to continually reference hardware-level connection state information and the hardware network protocol stack is configured to continually reference kernel-level connection state information and socket layer-level connection state information" recited in independent claim 1. Reading out an explicit limitation in the claims is clearly contrary to well established precedent that "[the Examiner] must consider all claim limitations when determining patentability of an invention over the prior art." *In re Gulack*, 703 F.2d 1381, 1385 (Fed. Cir. 1983).

Turning to the cited portions of Boucher, Fig. 22 of Boucher is directed to a receive sequencer. Receive sequencers are first introduced in the specification of Boucher by way of Fig. 13 as features located within the INIC/CPD. *See* Boucher: Fig. 13 (elements 418-424); and paragraphs [0084]-[0086]. Appellants respectfully assert that a receive sequencer is merely a classifier operating in tandem with a processor for purposes of analyzing and managing incoming packets. As a result of analyzing an incoming packet, the receive sequencer is able to route the packet to the proper queue. *See* Boucher: paragraphs [0099]-[0100] and [0941]. Appellants respectfully assert that, because receive sequencers are entirely native to the INIC/CPD, Fig. 22 cannot be properly construed to disclose or render obvious a software network protocol stack. Further, because receive sequencers strictly pertain to the process of receiving a packet in the INIC/CPD and routing the packets to the proper queue and processor features native to the INIC/CDP, Fig. 22 does not disclose or render obvious the storage of hardware-level connection

state information. In addition, Fig. 22 does not disclose or render obvious a hardware network protocol stack for reasons previously discussed in Section (VI)(A).

With respect to Fig. 23 of Boucher, Fig. 23 is directed to the transfer of fast-path eligible packets from the INIC/CPD to the host. *See* Boucher: paragraph [0133]. At best, Fig. 23 of Boucher depicts a correlation between the data by the INIC/CPD (*i.e.*, elements 2308, 2313, 2316, etc. as the data received as part of a multi-packet message) and the data once it is stored on the host (*i.e.*, cached "DATA" blocks shown within element 2311). Because Fig. 23 neither discloses nor renders obvious a hardware network protocol stack nor a software network protocol stack, it cannot further disclose or render obvious the limitation of "wherein after the hardware network protocol stack accepts transfer of the network connection the software network protocol stack is configured to continually reference hardware-level connection state information and the hardware network protocol stack is configured to continually reference kernel-level connection state information and socket layer-level connection state information."

Finally, Fig. 8 of Boucher is directed to the various sequencer infrastructure contained within the INIC/CPD. *See* Boucher: paragraphs [0067] – [0068]. Because the features disclosed in Fig. 8 are entirely native to the INIC/CPD, Fig. 8 cannot be properly construed to disclose or render obvious the limitation of "wherein after the hardware network protocol stack accepts transfer of the network connection the software network protocol stack is configured to continually reference hardware-level connection state information and the hardware network protocol stack is configured to continually reference kernel-level connection state information and socket layer-level connection state information" for at least the same or similar reasons discussed above in relation to Fig. 22.

C. Summary

In view of the above, the Examiner has failed to show the presence of all elements in the prior art and thereby has failed to satisfied the requirements of *KSR International Co.,* which requires that the Examiner "articulate the following: (1) a finding that the prior art included each element claimed, although not necessarily in a single prior art reference, with the only difference between the claimed invention and the prior art being the lack of actual combination of the elements in a single prior art reference; ..." MPEP § 2143(A) citing *KSR International Co. v. Teleflex Inc.*, 127 S.Ct. 1727, 1739, 75 U.S.L.W. 4289 (2007). Accordingly, the Examiner has failed to show sufficient evidence of *prima facie* obviousness.

VIII. CONCLUSION

In view of the above, as the Examiner has failed to show sufficient evidence for a *prima facie* to support prima facie obviousness, the Appellants have carried their burden in showing that the Examiner erred in finally rejecting the claims. *In re Kahn*, 441 F.3d 977, 985-986 (Fed. Cir. 2006) ("On appeal to the Board, an applicant can overcome a rejection by showing insufficient evidence of *prima facie* obviousness or by rebutting the *prima facie* case with evidence of secondary indicia of nonobviousness") (emphasis in original) (quoting *In re Rouffet*, 149 F.3d 1350, 1355 (Fed. Cir. 1998)); *see also* 37 C.F.R. § 41.37(c)(1)(vii). Favorable consideration of the present application is respectfully requested.

Dated: May 18, 2009

Respectfully submitted,

By /Robert P. Lord/

Robert P. Lord
Registration No.: 46,479
OSHA · LIANG LLP
909 Fannin Street, Suite 3500
Houston, Texas 77010
(713) 228-8600
(713) 228-8778 (Fax)

523950_2

APPENDIX A

Claims Involved in the Appeal of Application Serial No. 10/698,212

1. A method of processing a network connection in a computer system, comprising:
   establishing the network connection by a software network protocol stack implemented in the kernel of an operating system associated with the computer system, wherein the kernel maintains kernel-level connection state information for the network connection, and wherein a socket layer maintains socket layer-level connection state information for the network connection;
   determining whether to offload the network connection from the software network protocol stack to a hardware network protocol stack implemented in a TCP Offload Engine (TOE)-capable network interface card operatively connected to the computer system;
   transferring the network connection from the software network protocol stack to the hardware network protocol stack using a network interface card driver when it is determined to offload the network connection from the software network protocol stack to the hardware network protocol stack; and
   determining to accept the transfer of the network connection at the hardware network protocol stack based on a processing capability of the hardware network protocol stack, wherein the network interface card maintains hardware-level connection state information for the network connection, and
   wherein after the hardware network protocol stack accepts transfer of the network connection the software network protocol stack is configured to continually reference hardware-level connection state information and the hardware network protocol stack is configured to continually reference kernel-level connection state information and socket layer-level connection state information.

2. The method as recited in claim 1, wherein transferring the network connection from the software network protocol stack to the hardware network protocol stack using the network interface card driver comprises:

obtaining, by the network interface card driver for the software network protocol stack, a hardware connection identifier maintained by the network interface card in association with hardware-level connection state information for the network connection, wherein the software network protocol stack is configured to use the hardware connection identifier to obtain hardware-level connection state information for the network connection; and

obtaining, by the network interface card driver for the hardware network protocol stack, a reference to socket layer-level connection state information and a reference to kernel-level connection state information, wherein the hardware network protocol stack is configured to use the references to create mappings from the hardware connection identifier to both socket layer-level connection state information and kernel-level connection state information, wherein the hardware network protocol stack is further configured to use the mappings to obtain kernel-level connection state information and socket layer-level connection state information for the network connection.

3. The method as recited in claim 1, wherein determining whether to offload the network connection is performed by the operating system kernel of the computer system.

4. The method as recited in claim 3, wherein determining whether to offload the network connection is performed by the socket layer of the operating system kernel.

5. The method as recited in claim 1, wherein determining whether to offload the network connection is performed by the software network protocol stack.

6. (Canceled)

7. (Canceled)

8. (Canceled)

9. The method as recited in claim 1, wherein the hardware network protocol stack is capable of determining whether to offload the network connection back to the software network protocol stack.

10. The method as recited in claim 9, further comprising:

receiving an indicator from the hardware network protocol stack, the indicator indicating a request to transfer the network connection back to the software network protocol stack.

11. The method as recited in claim 10, further comprising:

obtaining the hardware-level connection state information for the network connection from the hardware network protocol stack using the network interface card driver and the hardware connection identifier for the network connection when the indicator is received; and

handling the network connection by the software network protocol stack using the obtained hardware-level connection state information.

12. (Canceled)

13. The method as recited in claim 11, further comprising:

obtaining at least one of unsent and undelivered data by the software network protocol stack from the hardware network protocol stack, thereby enabling the software network protocol stack to process the unsent or undelivered data.

14. (Canceled)

15. (Canceled)

16. (Canceled)

17. The method as recited in claim 9, further comprising:

handling the network connection by the software network protocol stack after the network connection is offloaded back to the software network protocol stack from the hardware network protocol stack.

18. (Canceled)

19. The method as recited in claim 1, further comprising:

handling the network connection by the software network protocol stack until it is determined to offload the network connection to the hardware network protocol stack.

20. (Canceled)

21. (Canceled)

22. (Canceled)

23. (Canceled)

24. The method as recited in claim 1, wherein the kernel-level connection state information comprises IP addresses and ports for a client and server of the network connection, and at least one of send and receive sequence numbers of one or more packets for the network connection.

25. The method as recited in claim 24, wherein the kernel-level connection state information further comprises:

a round trip estimate.

26. The method as recited in claim 25, wherein the kernel-level connection state information further comprises:

a congestion window and slow start information.

27. (Canceled)

28. The method as recited in claim 1, wherein upon transferring the network connection from the software network protocol stack to the hardware network protocol stack, the method further comprising:

sending one or more inbound packets by the hardware network protocol stack to the socket layer using the network interface card driver and receiving one or more outbound packets by the hardware network protocol stack from the socket layer using the network interface card driver, wherein the network interface card driver maintains a copy of each packet until the packet reaches its intended destination.

29. (Canceled)

30. An apparatus for processing a network connection in a computer system, comprising:

means for establishing the network connection by a software network protocol stack implemented in the kernel of an operating system associated with the computer system, wherein the kernel maintains kernel-level connection state information for the network connection, and wherein a socket layer maintains socket layer-level connection state information for the network connection;

means for determining whether to offload the network connection from the software network protocol stack to a hardware network protocol stack implemented in a TOE-capable network interface card operatively connected to the computer system;

means for transferring the network connection from the software network protocol stack to the hardware network protocol stack when it is determined to offload the network connection from the software network protocol stack to the hardware network protocol stack, and

means for determining to accept the transfer of the network connection at the hardware network protocol stack based on a processing capability of the hardware network protocol stack, wherein the network interface card maintains hardware-level connection state information for the network connection, and wherein after the hardware network protocol stack accepts transfer of the network connection the software network protocol stack is configured to continually reference hardware-

level connection state information and the hardware network protocol stack is configured to continually reference kernel-level connection state information and socket layer-level connection state information.

31. A computer-readable medium storing thereon computer-readable instructions for processing a network connection in a computer system, comprising:

instructions for establishing the network connection by a software network protocol stack implemented in the kernel of an operating system associated with the computer system, wherein the kernel maintains kernel-level connection state information for the network connection, and wherein a socket layer maintains socket layer-level connection state information for the network connection;

instructions for determining whether to offload the network connection from the software network protocol stack to a hardware network protocol stack implemented in a TOE-capable network interface card operatively connected to the computer system;

instructions for transferring the network connection from the software network protocol stack to the hardware network protocol stack using a network interface card driver when it is determined to offload the network connection from the software network protocol stack to the hardware network protocol stack; and

instructions for determining to accept the transfer of the network connection at the hardware network protocol stack based on a processing capability of the hardware network protocol stack, wherein the network interface card maintains hardware-level connection state information for the network connection, and wherein after the hardware network protocol stack accepts transfer of the network connection the software network protocol stack is configured to continually reference hardware-level connection state information and the hardware network protocol stack is configured to continually reference kernel-level connection state information and socket layer-level connection state information.

32. A network device comprising:

an operating system including a software network protocol stack implemented in the kernel of an operating system associated with the computer system, wherein the kernel maintains kernel-level connection state information for a network connection, and wherein a socket layer maintains socket layer-level connection state information for the network connection;

a hardware network protocol stack coupled to the software network protocol stack, wherein the hardware network protocol stack is implemented in a TOE-capable network interface card operatively connected to the computer system, wherein the operating system being configured for determining whether to offload the network connection to the hardware network protocol stack and transferring the network connection from the software network protocol stack to the hardware network protocol stack using a network interface card driver when it determines that it will offload the network connection to the hardware network protocol stack; and

a control component being configured for determining to accept the transfer of the network connection at the hardware network protocol stack based on a processing capability of the hardware network protocol stack and wherein the network interface card maintains hardware-level connection state information for the network connection, and wherein after the hardware network protocol stack accepts transfer of the network connection the software network protocol stack is configured to continually reference hardware-level connection state information and the hardware network protocol stack is configured to continually reference kernel-level connection state information and socket layer-level connection state information.

33. The network device as recited in claim 32, wherein the software network protocol stack is a TCP/IP stack and the hardware network protocol stack is a TCP/IP stack.

34. (Canceled)

35. (Canceled)

36. The network device as recited in claim 32, wherein the hardware network protocol stack is capable of determining whether to offload the network connection back to the software network protocol stack.

37. The network device as recited in claim 36, wherein the hardware network protocol stack sends an indicator when it requests to transfer the network connection back to the software network protocol stack.

38. The network device as recited in claim 37, wherein the software network protocol stack is adapted for obtaining hardware-level connection state information for the network connection from the hardware network protocol stack using the network interface card driver and the hardware connection identifier for the network connection when the hardware connection indicator is received, thereby enabling the software network protocol stack to handle the network connection using the obtained hardware-level connection state information.

39. (Canceled)

40. The network device as recited in claim 38, wherein the software network protocol stack is further adapted for obtaining at least one of unsent and undelivered data from the hardware network protocol stack, thereby enabling the software network protocol stack to process the unsent or undelivered data.

41. (Canceled)

42. (Canceled)

43. (Canceled)

44. (Canceled)

45. The network device as recited in claim 36, wherein the software network protocol stack is capable of handling the network connection when the network connection is offloaded back to the software network protocol stack from the hardware network protocol stack.

46. (Canceled)

47. The network device as recited in claim 32, wherein the software network protocol stack handles the network connection until it is determined by the operating system to offload the network connection to the hardware network protocol stack.

48. (Canceled)

49. (Canceled)

50. (Canceled)

51. (Canceled)

52. The network device as recited in claim 50, wherein the kernel-level connection state information comprises IP addresses and ports for a client and server of the connection, and at least one of send and receive sequence numbers of one or more packets for the connection.

53. The network device as recited in claim 52, wherein the kernel-level connection state information further comprises:

a round trip estimate.

54. The network device as recited in claim 53, wherein the kernel-level connection state information further comprises:

a congestion window and slow start information.

55. (Canceled)

56. The network device as recited in claim 32, wherein upon transferring the network connection from the software network protocol stack to the hardware network protocol stack, the hardware network protocol stack is in communication with a socket layer of the software network protocol stack, thereby enabling data to be sent by the hardware network protocol stack to the socket

layer and enabling data to be received by the hardware network protocol stack from the socket layer.

57. The method as recited in claim 1, wherein the network interface card driver is configured to maintain a copy of kernel-level connection state information and a copy of socket layer-level connection state information after the hardware network protocol stack accepts transfer of the network connection.

58. The network device as recited in claim 32, wherein the network interface card driver is configured to maintain a copy of kernel-level connection state information and a copy of socket layer-level connection state information after the hardware network protocol stack accepts transfer of the network connection.

APPENDIX B

No evidence pursuant to §§ 1.130, 1.131, or 1.132 or entered by or relied upon by the examiner is being submitted.

APPENDIX C

No related proceedings are referenced in II. above, hence copies of decisions in related proceedings are not provided.